GENERIC SORT 알고리즘

- **generic swap**의 원리를 이해한다.
- **generic sort**의 원리를 이해한다.

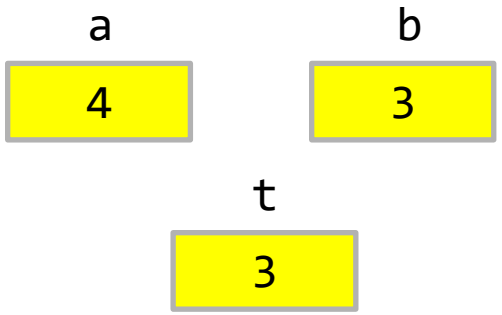
1) generic swap의 구현
2) generic sort의 구현

swap 알고리즘

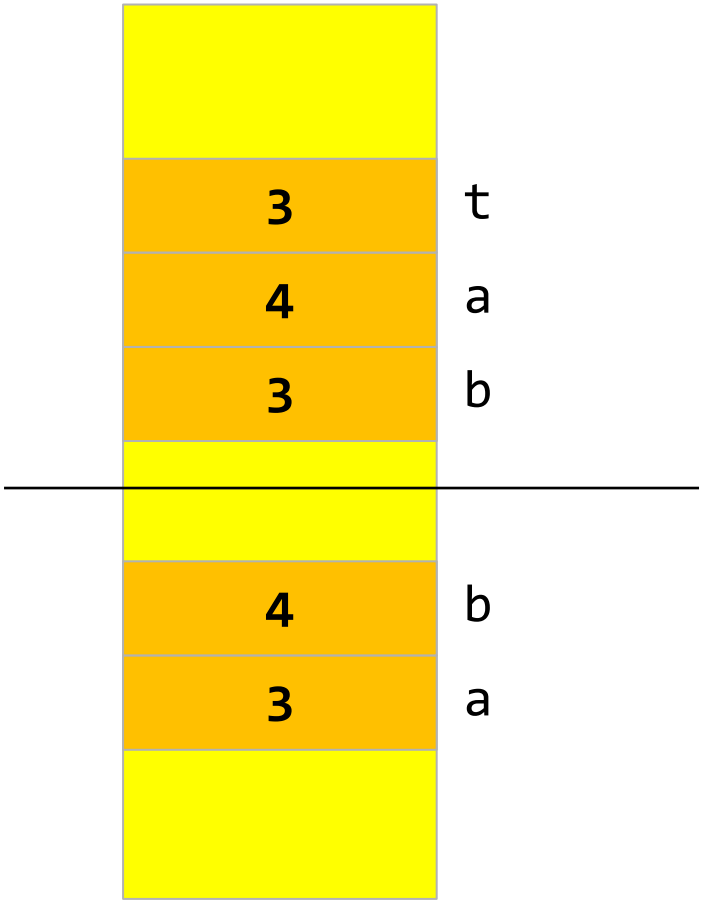
```
int main()
{
    int a=3, b=4;
    int t;

    t = a;
    a = b;
    b = t;

    printf( "a=%d, b=%d\n", a, b );
    return 0;
}
```



swap 알고리즘



generic swap 알고리즘

```
void swap ( void *a, void *b, int size )
{
    int i;
    char t;
    char *p = (char*)a;
    char *q = (char*)b;
    for( i=0 ; i < size; i++ )
    {
        t = p[i];
        p[i] = q[i];
        q[i] = t;
    }
}
```

```
void generic_swap(void *a, void *b,
 int size)
{
    char t;

    do {
        t = *(char *)a;
        *(char *)a++ = *(char *)b;
        *(char *)b++ = t;
    } while (--size > 0);
}
```

bubble sort 알고리즘

```
void bubble( int *a, int n )
{
    int i,j;
    for(i=0; i<n-1; i++ )
        for(j=0; j<n-1-i; j++ )
            if( a[j] > a[j+1] )
                generic_swap( a+j, a+j+1, sizeof(a[0]));
}
```

bubble sort 알고리즘 일반화

```
void bubble_sort ( void *a, int n, int width )
{
    int i,j;

    for( i=0; i<n-1; i++ )
        for( j=0; j<n-1-i; j++ )
            if( *(int*)((char*)a+j*width) > *(int*)((char*)a+(j+1)*width) )
                generic_swap( (char*)a+j*width, (char*)a+(j+1)*width, width );
}
```

문제점 : 값이 필요한 알고리즘은 타입을 일반화 할 수 없다.
해결책 : 라이브러리가 일반화 할 수 없다면 유저에게 위임하자.

bubble sort 알고리즘 일반화

```
void bubble( void *a, int n , int size,
            int (*cmp)(const void*,const void*))
{
    int i,j;
    for(i=0; i<n-1; i++ )
        for(j=0; j<n-1-i; j++ )
            if( cmp( (char*)a+j*size,(char*)a+(j+1)*size) )
                generic_swap( (char*)a+j*size,
                              (char*)a+(j+1)*size,
                              size);
}
//-------------------------------------------------------------------

int int_cmp(const void* a,const void* b)
{
    return  (*(int*)a - *(int*)b) > 0;
}
```

kernel sort 알고리즘 일반화

```
void sort(void *base, size_t num, size_t size,
      int (*cmp_func)(const void *, const void *),
      void (*swap_func)(void *, void *, int size))
{
    /* pre-scale counters for performance */
    int i = (num/2 - 1) * size, n = num * size, c, r;
    if (!swap_func)
        swap_func = (size == 4 ? u32_swap : generic_swap);
 /* heapify */
    for ( ; i >= 0; i -= size) {
        for (r = i; r * 2 + size < n; r  = c) {
            c = r * 2 + size;
            if (c < n - size &&
                    cmp_func(base + c, base + c + size) < 0)
                c += size;
            if (cmp_func(base + r, base + c) >= 0)
                break;
            swap_func(base + r, base + c, size);
        }
    }
```

kernel sort 알고리즘 일반화

```
 /* sort */
    for (i = n - size; i > 0; i -= size) {
        swap_func(base, base + i, size);
        for (r = 0; r * 2 + size < i; r = c) {
            c = r * 2 + size;
            if (c < i - size &&
                    cmp_func(base + c, base + c + size) < 0)
                c += size;
            if (cmp_func(base + r, base + c) >= 0)
                break;
            swap_func(base + r, base + c, size);
        }
    }
}
```